REMARKS

In view of the foregoing amendments and following remarks responsive to the Office Action dated June 6, 2008, Applicants respectfully request favorable reconsideration of this Application.

Applicants respectfully thank the Office for the withdrawal of all previous rejections. However, the Office has proffered new rejections of all pending claims, claims 1-35. Particularly, the Office has rejected claims 1-2, 5-22, 24, and 26-35 under 35 U.S.C. 103(a) as unpatentable over Bailey in view of Kodosky and claims 3, 4, 23, and 25 under 35 U.S.C. 103(a) as unpatentable over Bailey in view of Kodosky and further in view of Visio 2000.

The present invention is a method and software product for defining, representing, and/or documenting object-oriented programming applications, and particularly, those developed to work in a graphical user interface. The invention represents the applications in one or more diagrams termed "Object and Event Diagrams" ("OEDs"). The technique allows an application architect to define the program basis and the program logic without concern about details of the user interface, such as how actions will be executed (which can be determined by the programmer).

Bailey discloses a program development environment for use in generating application programs. Kodosky discloses a system for creating and using configuration diagrams for configuring distributed systems, including creating programs, managing programs, deploying programs, and configuring

remote execution or inter-operation of distributed programs, and executing distributed applications.

Applicants respectfully traverse insofar as the parallel references do not teach that for which they have been cited. Independent claim 1 is reproduced below for reference.

1. (Previously Presented) A method for graphically representing object oriented programming logic, the method comprising the steps of:

(1) providing a plurality of different symbols for use in a diagram of object oriented programming logic, each different symbol representing a different type of object in object oriented programming;
(2) selecting an object as a main object of the logic to be represented in the diagram;
(3) drawing a symbol corresponding to the main object and labeling the symbol with a label descriptive of the object's features so that it is distinguishable from other symbols of the same object type;
(4) for each object assigned to or defined within the main object, drawing a symbol corresponding to that object and labeling the symbol with a label descriptive of the object's features; and
(5) drawing a line between each object drawn in step (4) and another object in the graphical representation to which it is assigned or within which it is defined.

The Office asserts that Bailey teaches in column 4, lines 23-26, step (5) of claim 1 of drawing a line between each object drawn in step (4) and another object in the graphical representation to which it is assigned or within which it is defined.

This is not accurate. Rather, column 4, lines 23-26 of Bailey state:

> According to the invention, these symbols can be linked together by the developer in the form of a data flow or block diagram that logically represents the flow of data and control information into, out of, and between the selected control objects.

This section of Bailey discloses nothing but a flow diagram, which shows the order of steps, or a block diagram, which shows the flow of data between blocks. This has essentially nothing to do with step (5) of claim 1. Specifically, step (5) of claim 1 does not recite simply drawing a line between objects, but drawing a line between objects to represent a very specific concept, namely, assignment to or definition within another object in object oriented programming.

This concept is completely lacking in Bailey. Bailey, column 4, lines 23-26 discloses a flow chart or a block diagram. Particularly, in OOP, an object may be defined within another object or may be assigned to another object. The present invention provides a technique for graphically illustrating this concept in an overall system and method for representing object-oriented programming concepts. Column 4, lines 23-26 of Bailey does not relate to this concept, let alone disclose the specific features claimed in step (5) of claim 1.

Furthermore, the Office relies on Kodosky, paragraphs 11 and 12 as teaching step (2) of claim 1, namely, selecting an object as a main object of the logic to be represented in the diagram. The Office further asserts that it would have been obvious to combine this feature of Kodosky with Bailey because it would help users to easily understand logic that was described in detail according to the specification of the system.

However, the alleged motivation is taken directly from Applicants' specification, and not from any part of the cited prior art. Accordingly, it comprises impermissible hindsight reconstruction of the invention based on Applicants' disclosure.

Accordingly, Applicants respectfully request the Office to withdraw the rejection of claim 1 insofar as the prior art of record does not disclose all of the claim features and, furthermore, the proposed combination of features of different prior art references is based on impermissible hindsight reconstruction.

The dependent claims add substantial additional distinguishing patentable features. For instance, claim 3 depends from claim 1 and adds the step of graphically denoting the main object in the diagram by drawing another symbol around the symbol for the main object. The Office asserts that this is found in Kodosky, which teaches graphically denoting the main object in the diagram, and Visio, which teaches drawing another symbol around the symbol for the main object.

However, Visio teaches no such thing. In particular, the Office has simply cited a portion of the Visio reference that discloses a drawing program that would allow the user to draw a circle around another graphical representation. It does not disclose or suggest drawing a circle around a main object to denote it as the main object of a graphical representation of object-oriented programming. Thus, Visio discloses nothing but a tool that could be used to practice the disclosed and claimed feature of claim 3. It clearly does not disclose the feature itself.

Accordingly, claim 3 even further patentably distinguishes over the prior art of record.

Claim 4 depends from claim 3 and adds that step (7) comprises drawing a circle completely enclosing the symbol of the main object.

It should be apparent from discussion above of claim 3 that claim 4 distinguishes over the prior art for the same reasons discussed above in connection with claim 3.

Accordingly, claim 4 even further patentably distinguishes over the prior art of record.

Claim 7 depends from claim 1 and adds that the method is used to document software. The Office asserts that this is found in paragraph 1 of Kodosky. However, paragraph 1 of Kodosky teaches no such thing. Paragraph 1 of Kodosky states that his engine is used for specifying or creating distributed systems and/or applications, not documenting existing software.

Accordingly, claim 7 even further patentably distinguishes over the prior art of record.

Claim 9 depends from claim 1 and adds repeating steps 1-5 to create a plurality of diagrams corresponding to separate parts of an overall application and wherein a first object is the main object appearing in at least the first one of the diagrams and is not a main object appearing in at least a second one of the diagrams.

The Office asserted that this is taught in Bailey at column 8, lines 14-18.

Applicants respectfully traverse. Column 8, lines 13 through 18 of Bailey are reproduced below for ease of reference.

> As described below, the designer window 406 is configured to display a corresponding symbol for each program object added to the form window 404. These symbols, moreover, may be graphically linked together in order to create a data flow or a block

> diagram that logically represents the flow of data and/or execution control of the application program that is being developed.

This text does not appear to have anything to do with the subject matter of claim 9. This text discusses the creation of a single diagram and, therefore, cannot possibly disclose the subject matter of claim 9.

Accordingly, claim 9 even further patentably distinguishes over the prior art of record.

Claim 10 depends from claim 9 and adds that the second diagram does not disclose objects assigned to and defined within the first object and the first diagram does disclose objects assigned to and defined within the first diagram.

The Office asserts that this is found in Kodosky paragraph 15. Paragraph 15 of Kodosky is reproduced below for ease of reference.

> The configuration diagram may support various types of views, such as an entire system view, a subsystem view, a device view, a program view, etc. For example, the user can 'drill down' in the configuration diagram to view a selected portion of the diagram, e.g., a selected subsystem of devices, a single device, the program associated with a device, the data points associated with the device, the I/O channels associated with the device, etc.

Paragraph 15 of Kodosky does not appear to be relevant to claim 10. At best, it appears to disclose something that is essentially the opposite of what is claimed in claim 10. Specifically, claim 10 discusses two diagrams and recites that one of these diagrams does <u>not</u> contain certain information. Paragraph 15 of Kodosky, on the other hand, discusses a <u>single</u> diagram within which one can "drill down" to see detail. Thus, not only is it not seen how this discussion of a single

configuration diagram could teach something about the nature and relationship of two diagrams, it is substantially contrary to claim 10 insofar as all of the information discussed in this paragraph appears to be within a single configuration diagram.

Accordingly, claim 10 even further patentably distinguishes over the prior art of record.

Claim 12 depends from claim 10 and adds that the labels of the first object in the second diagram identifies the first diagram as disclosing further details of the first object.

The Office asserts that this is found in Bailey at column 9, lines 50-53. This portion of Bailey discusses the properties window 418 that displays the properties of a select program object residing in the form window 404. Particularly, Bailey includes a property window 422 divided into two columns, a name column and a current value column. Column 9, lines 50-53 disclose:

> The name column 422A identifies all of the properties associated with the program object selected in the object list 420, while the current value column 422B shows the values that are currently associated with those properties.

Accordingly, it discloses that one particular diagram, namely, property window 422 discloses information about objects in another window, namely, form window 404. However, while claim 12 does pertain to information about objects in one diagram appearing in another diagram, it actually is directed to an additional feature that is quite different than anything disclosed in Bailey. It recites that there is a label in one of the diagrams identifying another diagram as

disclosing further details of the first object. While this portion of Bailey discloses the existence of a second window disclosing details about objects in a first window, it has nothing to do with a label in the first window identifying the second window, which is what claim 12 is about.

Accordingly, claim 12 further patentably distinguishes over the prior art of record.

Claims 13, 14 and 16 recite that the symbols representing different object types include different symbols representing application-type objects, window-type objects, class-type objects, event script-type objects, method-type objects, etc.

The Office acknowledges that Bailey does not teach these particular symbols. Nevertheless, the Office rejected the claim, asserting that Bailey does teach a plurality of objects or icons representing text boxes, radio buttons, scroll bars, menu bars and so on, "each icon represent[ing] a corresponding object class that is available for use by developer". The Office further asserts that it would have been obvious to add more graphical icons representing other programming objects which will include the functionality of the system.

Applicants respectfully traverse. The Office's assertion that Bailey renders obvious any symbol representing any type of object is improper. Applicants' specific symbols and what they represent as recited in claims 13, 14 and 16 are not merely random or obvious design choices among which any skilled artisan might choose. Rather, the particular types of objects and other OOP programming entities that are both logically grouped together by virtue of being

assigned the same symbol and differentiated from each other by virtue of being assigned different symbols is a crucial aspect of the primary purpose of the invention, namely, a tool for representing object-oriented programming in a simple, yet powerful, way. The particular choice as to what types of objects are grouped together under a single symbol and what types of objects have different symbols is crucial to the effectiveness of the invention in achieving its primary goals of providing a simple, yet powerful, OOP representation tool.

When the specification of the present Application is read in its entirety, it should be absolutely clear that these particular symbols and what they represent provide enormous functionality in terms of simplifying both an OOP software developer's task of describing the desired code and an OOP programmer's task of understanding what is required because, inter alia, the representation in accordance with the present invention's scheme permit the developer to represent the program basis and logic without concern about programming-level details, such as how actions will be taken or which event actions will take place. See p. 4, ll. 1-10 of the present application.

By way of example, let us consider if, instead of a different symbol to represent application-type objections, window-type objects, class-type objects, event script-type objects, method-type objects, data transfer, databases, remote links, and inheritance as claimed in claim 14, the present invention had different symbols to represent text boxes, radio buttons, scroll bars, menu bars, and so on as allegedly disclosed in Bailey. If such substitutions were made, the present invention would be utter nonsense (or, at least, a completely different invention),

thus showing that the particular choices made as to what the symbols represent is far from a mere design choice of no patentable weight. In fact, there are significant features of the invention.

Let us not forget that the invention is a method and apparatus to visually represent something in an efficient and effective way. To assert without support that the particular ways Applicant chooses to represent such things is obvious is intellectually dishonest. These differences over the prior art are exactly what makes the invention innovative.

Accordingly, claims 13, 14, and 16 further patentably distinguishes over the prior art of record.

Claim 15 depends from claim 14 and adds that the data transfer, remote link, and inheritance symbols are drawn connecting two other object symbols.

Again, similar to its analysis of claims 13, 14 and 16, the Office essentially asserts that this is found in Bailey because Bailey teaches "wires" that connect different symbols. However, again, this analysis is faulty. Claim 15 does not simply recite that there are symbols that connect other symbols. Rather, claim 15 depends from claim 14, which depends from claim 2, which depends from claim 1, which collectively recite very specific symbols having very specific meanings. To assert that the prior reference merely need show wires connecting two symbols teaches all of the rich detail in these claims is to utterly ignore most of the claim language.

Claim 18 depends from claim 13 and recites that the event script-type object is drawn connected to another object that directly executes the event

script corresponding to the event script symbol. The Office asserts that this is found in column 9, lines 60-62 of Bailey.

Again, this portion of Bailey discusses the use of wires between symbols (in order to create a data and/or execution control flow diagram).

Applicants respectfully traverse. The Office appears to be misinterpreting claim 18, which does not pertain to connecting two objects with another object. Rather, claim 18 recites that the event script-type object is connected to another object that directly executes the corresponding event script. It does not discuss the event script object connecting the "another object" to yet a further object. In the context of an event script-type object, such connection probably would not make much sense in most, if not all, cases. Furthermore, as previously discussed in connection with some of the claims discussed above, this portion of Bailey does not appear to disclose anything other than drawing wires between symbols in the nature of a flow chart or block diagram.

Accordingly, claim 18 further patentably distinguishes over the prior art of record.

Claim 19 depends from claim 13 and adds that the method object symbol is drawn connected to the main object of the diagram and represents that the object is available within that main object and does not represent that the main object invokes it. The Office asserted that this is found in Bailey at column 7, lines 57-60 and column 8, lines 14-18. Bailey, column 7, lines 57-60 state "Each icon represents a corresponding component control or program object class that is available for use by the developer in creating application programs" and

column 8, lines 14-18 states "These symbols, moreover, may be graphically linked together in order to create a data flow or block diagram that logically represents the flow of data and/or execution control of the application program that is being developed."

Applicants respectfully traverse. Once again, the Office is taking a very generic disclosure of a flow diagram or a block diagram and asserting that it teaches the very specific limitations of a claim. Claim 19 does not simply recite connecting two objects. Rather, it recites connecting the main object of the diagram using a very specific symbol representing a very specific type of object and indicating that it means something very specific and does not mean something else that is very specific. The cited portions of Bailey simply do not disclose all of these specific recitations. In fact, the Office has already admitted that Bailey does not even teach either a main object or a method-type object symbol. Accordingly, it is not seen how it could possibly teach a method-type object connected to a main object. Furthermore, it certainly does not teach that such a connection represents that the object is available within that main object and does not represent that the main object invokes it.

Accordingly, claim 19 even further patentably distinguishes over the prior art of record.

Claims 22 through 35 are computer product claims corresponding to some extent to method claims 1-21 discussed above.

Particularly, independent computer product claim 22 has been amended to correspond substantially to claim 1 with respect to the afore-discussed

recitations that distinguish over the prior art of record. Dependant claim 23 corresponds substantially to the further distinguishing features discussed above in connection with claim 3. Dependent claim 25 recites substantially the same further distinguishing features as claim 4 discussed above. Dependent computer product claims 30, 31, 32, 33, and 34 recite additional distinguishing subject matter substantially similar to dependent method claims 13, 14, 15, 16 and 17, respectively. Therefore, those claims even further distinguish over the prior art at least for all of the reasons set forth above in connection with the corresponding method claims.

Furthermore, dependent claim 27 depends from claim 22 and adds that the sixth set of instructions enables the user to prepare a plurality of diagrams corresponding to several parts of an overall application and further comprising computer-readable instructions for enabling the user to specify relationships between individual ones of the diagrams.

The Office asserts that this is found in Bailey at column 8, lines 14-18 and column 9, lines 53-57. However, these portions of Bailey (previously reproduced above) merely disclose that one can create a single diagram comprising multiple figures linked together in order to create a data flow or block diagram. As discussed above in connection with claim 9, for instance, this really does not have anything to do with what is being claimed in claim 27 pertaining to displaying the relationships between multiple diagrams.

Claim 28 depends from claim 27 and further embellishes the sixth instructions, reciting that they enable the user to include references associated

with symbols in one diagram identifying at least one other diagram within which the object represented by the symbol also appears.

The Office asserts that this is disclosed in Kadosky, paragraph 15. However, as discussed above in connection with claim 9, paragraph 15 of Kadosky. Actually appears to teach something substantially the opposite of what is claimed here. Particularly, Kadosky paragraph 15 discusses a single diagram that the user can "drill down" into in order to view a selected portion of the diagram. Claim 28, on the other hand, discusses a plurality of diagrams corresponding to separate parts of an overall application and specifying relationships between individual ones of the diagrams.

Accordingly, claims 27 and 28 even further patentably distinguish over the prior art of record.

Conclusion

Applicants have amended other claims in addition to claim 22 in order to correct clerical or typographical type errors as well as to make the claim language consistent with newly amended claim 22.

In view of the foregoing remarks, this application is now in condition for allowance. Applicants respectfully request the Office to issue a Notice of Allowance at the earliest possible date. The Examiner is invited to contact Applicants' undersigned counsel by telephone call in order to further the prosecution of this case in any way.

Respectfully submitted,

Dated: August 19, 2008

/Theodore Naccarella/

Theodore Naccarella, Reg. No. 33,023
Saul Ewing LLP
Centre Square West
1500 Market Street
Philadelphia, PA 19102

Tele: (215) 972-7877
Fax: (215) 972-4161

Attorneys for Applicants